Rapport de stage

Stage effectué à l'ENS de Lyon

Laboratoire de l'Informatique du parallélisme (LIP)

pour l'obtention du diplôme de Master

Étude et mise en œuvre d'un support pour la gestion des grandes données au sein de l'intergiciel DIET sur environnements applicatifs dédiés

Auteur :
Patrick Telemaque

Responsable :
M. Eddy Caron

29 Août 2014

Table des matières

Remerciements 5
Résumé 6

1 Introduction **10**
1.1 Contexte scientifique et industriel 10
1.2 Description de la plate-forme expérimentale 11
1.3 Problématiques et objectifs 12
1.4 Contribution 13

2 État de l'art **14**
2.1 Les protocoles de transfert de données 14
2.1.1 Expedat 15
2.1.2 Bitspeed 15
2.1.3 Aspera 15
2.1.4 Conclusion 17
2.2 Les modélisations de transfert de donneés 18
2.2.1 Modèle de Hockney 18
2.2.2 Famille de modèle LogP 18
2.2.2.1 Modèle LogP 19
2.2.2.2 Modèle pLogP 19
2.2.2.3 Modèle LogGP 20
2.2.3 Conclusion 21

3 Expérimentations **23**
3.1 Performances des protocoles de transfert 23
3.1.1 Transfert entre des nœuds de site différent 23

3.1.2 Transfert entre des serveurs NFS de site différent . . . 23
3.1.3 Transfert entre un nœud et un serveur NFS de site différent . . . 24
3.1.4 Transfert entre le serveur graal et un serveur NFS . . 25
3.2 Performances des modèles et techniques de mesure de transfert de donneés . . . 26
3.2.1 Description des expériences . . . 26
3.2.1.1 LogP/MPI 1.3 . . . 27
3.2.1.2 Logp_multitest . . . 27
3.2.1.3 Les commandes utilisées . . . 28
3.2.2 Les expériences sur taurus-11.lyon.grid5000.fr . . . 29
3.2.3 Les expériences sur pastel-73.toulouse.grid5000.fr . . . 29

4 Modélisation **31**
4.1 Modélisation du temps de transfert de données . . . 31
4.1.1 Définition des variables . . . 31
4.1.2 Modèle . . . 32
4.1.3 Présentation des résultats . . . 33
4.1.3.1 Résultat obtenu pour le protocole Aspera . . 34
4.1.3.2 Résultat obtenu pour le protocole Expedat . 36
4.1.3.3 Résultat obtenu pour le protocole Bitspeed . 37
Conclusion générale . . . 38
Références . . . 39

Table des figures

1.1 Plate-forme expérimentale 11

2.1 Comparaison des protocoles de transfert 16

3.1 Temps de transfert entre 2 nœuds de site différent 24
3.2 Temps de transfert entre 2 serveurs NFS de site différent . . . 24
3.3 Temps de transfert entre un nœud et un serveur NFS de site différent 25
3.4 Temps de transfert entre le serveur graal et un serveur NFS . 25
3.5 Le modèle LogP 27
3.6 Performance réseau du modèle logP 29

4.1 Temps de transfert avec Aspera sur taurus-11.lyon.grid5000.fr 35
4.2 Temps de transfert avec Aspera sur pastel-73.toulouse.grid5000.fr 35
4.3 Temps de transfert avec Expedat sur taurus-11.lyon.grid5000.fr 36
4.4 Temps de transfert avec Expedat sur pastel-73.toulouse.grid5000.fr 36
4.5 Temps de transfert avec Bitspeed sur taurus-11.lyon.grid5000.fr 37
4.6 Temps de transfert avec Bitspeed sur pastel-73.toulouse.grid5000.fr 37

Liste des tableaux

2.1 Caractéristiques des protocoles 16
2.2 *Le modèle LogGP exprimé en fonction de pLogP* 21

3.1 Table des mesures sur le nœud de Lyon 30
3.2 Table des mesures sur le nœud de Toulouse 30

4.1 Tableau des variables du modèle 31
4.2 Table des paramètres pour taurus-11.lyon.grid5000.fr 34
4.3 Table des paramètres pour pastel-73.toulouse.grid5000.fr . . 34
4.4 Table des paramètres pour les protocoles 34

Remerciements

Mes remerciements s'adressent en premier lieu au grand architecte de l'univers, Dieu, sans qui rien de ceci ne serait possible.

Je veux remercier de façon très particulièrement chaleureuse mon maître de stage, Monsieur Eddy Caron, pour sa confiance, ses conseils et sa disponibilité qui m'ont permis de progresser sans cesse durant ces 6 mois de stage.

Je tiens également à remercier tous les membres de l'équipe Avalon du LIP, pour leur amitié, leur soutien et leur accueil dans le groupe.

Mes pensés vont également à tout le personnel et enseignants de l'Institut de la Francophonie pour l'Informatique (IFI), très spécialement Messieurs Victor Moraru, Nguyen Hong Quang et Ho Tuong Vinh pour leur conseil et le suivi qu'ils m'ont accordés pendant mes études de master et mon séjour au Vietnam.

Mille mercis à tous mes amis étudiants de l'IFI avec qui j'ai passé de bons moments pendant les périodes de stress, et qui, à leur façon m'ont donné la force d'avancer : Paterne, Selain, Landy, Farida, Youssouf, Mapenda, Hoa et j'en passe forcément.

J'exprime ma profonde gratitude à l'égard de ma grande famille pour leurs encouragements, leurs prières et leurs soutiens, qui malgré la distance n'a jamais cessé de me prêter main forte.

Finalement, merci à tous ceux qui ont croisé mon chemin à Hanoï et à Lyon, d'une façon ou d'une autre vous avez forcément influencé ce travail.

Résumé

Animérique est un projet dont l'objectif est de concevoir et déployer une plate-forme de calcul distribuée sur ressources hétérogènes. Cette plate-forme sera dédiée à l'industrie de l'animation. Ce projet est né de la rencontre entre deux mondes : la recherche sur la communauté du calcul haute performance par les chercheurs de l'Inria et de l'ENS de Lyon et la communauté d'animation à travers la société CapRézo.

Les besoins de calcul numérique se développent chaque année. Cependant les artistes peuvent bénéficier de certains outils efficaces pour la création et la modélisation, mais il y a un manque d'outils pour distribuer les tâches de calcul sur des plate-forme distribuées et hétérogènes. Encore aujourd'hui, certains studios distribuent presque manuellement les tâches. Certaines solutions existent en utilisant le paradigme Cloud, mais le modèle économique conduit à être dépendant d'un fournisseur et/ou implique d'envoyer des données critiques en dehors du territoire.

Un des problèmes majeures pour le partage des ressources et le calcul distribué est la gestion de données en environnement distribué. Chaque application a des besoins propres en terme d'accès ou production de données : grandes quantités de petites données de quelques kilooctets, ou données de plusieurs teraoctets. Dès lors que l'on utilise des plates-formes de calcul hétérogènes et distribuées, aussi bien les ressources de stockage (RAM, disque local ou distant, etc.) que les liens réseaux sont très discordants en terme de performance et de taille. Il convient donc d'adapter les politiques de déplacements, réplication et positionnement des données en fonction des besoins des applications, et des possibilités de la plate-forme sous-jacente.

Ce travail compare les approches commerciales de transport de données rapides à travers des Wide Area Network (WAN) à haut débit . Des solutions courantes, tels que le protocole de transport de fichiers (FTP) basé sur la pile TCP/IP, sont de plus en plus remplacées par des protocoles modernes basés sur des piles plus efficaces. Pour évaluer les capacités des applications actuelles pour le transport rapide des données, les solutions commerciales suivantes ont été étudiées : Aspera, Bitspeed et Expedat. Le but de ce travail est de tester les solutions dans les mêmes conditions réseaux et ainsi comparer les performances de transmission des dernières solutions propriétaires alternatives pour FTP/TCP dans les réseaux WAN à haut débit où il y a des latences élevées. Cette recherche porte sur une comparaison des approches utilisant des paramètres intuitifs tels que le taux de données et la durée de transmission.

La validation et la mise en œuvre des solutions conçues par le projet Animérique seront effectués en utilisant l'intergiciel DIET. Cet intergiciel est un logiciel développé par l'équipe Avalon (Inria / ENS de Lyon).

Mots clés : *transferts de données à haut débit , cloud computing, big data, protocole de transport.*

Abstract

Animerique is a project whose goal is to design and deploy a platform for distributed computing on heterogeneous resources. This platform will be dedicated to the animation industry. This project is born from the encounter between two worlds : the research community for high performance computing by researchers at INRIA and ENS Lyon, and community animation through the company CapRézo.

Needs numerical calculation grow each year. However, artists can benefit from some effective tools for creating and modeling, but there is a lack of tools to distribute computing tasks on distributed and heterogeneous platform. Even today, some studios almost manually distribute tasks. Some solutions exist using the cloud paradigm, but the economic model leads to be dependent on a supplier and / or involves sending critical data outside the territory.

One of the major problem for resource sharing and distributed computing is the data management in a distributed environment. Each application has its own needs in terms of access or data production : large quantities of small data of few kilobytes or terabytes of data. As long as that we use platforms heterogeneous and distributed computing, both storage resources (RAM, local or remote disk, etc..) and the network links are very discordant in terms of performance and size. It is therefore necessary to adjust movement policies, replication and data placement based on application needs and opportunities of the underlying platform.

This work compares commercial fast data transport approaches through high-speed Wide Area Network (WAN). Common solutions, such as File Transport Protocol (FTP) based on TCP/IP stack, are being increasingly replaced by modern protocols based on more efficient stacks. To assess the capabilities of current applications for fast data transport, the following commercial solutions were investigated : Aspera, Bitspeed and Expedat. The goal of this work is to test solutions under equal network conditions and thus compare transmission performance of recent proprietary alternatives for FTP/TCP within high-speed networks where there are high latencies in WANs. This research focuses on a comparison of approaches using intuitive parameters such as data rate and duration of transmission.

Validation and implementation of designed solutions by the project Animerique will be done using the DIET middleware. This middleware is a software developed by the Avalon team (Inria / ENS de Lyon).

Key words : *high-speed data transport, cloud computing, big data, transport protocol.*

Première partie

Chapitre 1

Introduction

1.1 Contexte scientifique et industriel

Ce stage s'inscrit dans le cadre du projet collaboratif Animérique entre l'Inria, l'ENS de Lyon et la société CapRézo.

L'Institut National de Recherche en Informatique et en Automatique (INRIA) est un organisme public de recherche français. Son objectif est de mettre en réseau les compétences de la recherche française dans le domaine des sciences et technologies de l'information.

L'École Normale Supérieure de Lyon (ou ENS de Lyon) est une grande école scientifique et littéraire française, l'une des quatre écoles normales supérieures. Elle forme à l'enseignement et à la recherche dans le domaine des sciences fondamentales et expérimentales ainsi que dans celui des lettres et sciences humaines.

À travers le projet Animérique, l'Inria, l'ENS et CapRézo joignent leurs efforts afin de concevoir et déployer une plate-forme de calculs distribués qui sera dédiée à l'industrie de l'animation. Les principaux thèmes abordés par le projet Animérique concernent la distribution des tâches de calcul sur des plates-formes distribuées et hétérogènes, l'accès et la gestion locale des ressources (ex : cluster ou Cloud), la planification des tâches et l'optimisation du temps de transfert des grands volumes de données. Ces solutions seront mises en œuvre en utilisant l'intergiciel DIET qui est un logiciel développé par l'équipe Avalon (Inria / ENS de Lyon).

Ces recherches sur les grappes et grilles sont au cœur des thématiques du laboratoire LIP de L'ENS de Lyon, en particulier sur des aspects comme l'évaluation de performance et sur la gestion de ressources dynamiques.

1.2 Description de la plate-forme expérimentale

Dans ce travail, les expériences ont été réalisées sur la grille de calcul française Grid'5000 [1]. L'environnement expérimental est constitué de deux sites séparés par un réseau étendu (WAN), un serveur de stockage et un poste client. La figure 1.1 représente l'architecture de cet environnement paramétré par n, qui constitue le nombre de nœuds. Nous utilisons le même nombre de nœuds sur les deux sites S_1 et S_2. $N_{i,j}$ est le nœud j du site i. La puissance des nœuds est différente d'un site à l'autre et peut grandement faire varier les performances selon que l'on se trouve sur des machines avec 2 ou 8 cœurs par exemple mais les comparaisons sont effectuées de manière unitaire, c'est-à-dire avec les mêmes machines. *bdp* représente la capacité du lien longue distance à 1 ou 10 Gbit/s. Le RTT varie selon les sites choisis (9,9 s entre Lyon et Toulouse). Les nœuds sont reliés par des cartes Ethernet à 1 Gbit/s. Chaque site possède un serveur NFS commun à tous les nœuds. Le serveur graal est un serveur qui se trouve à l'extérieur de la grille de calcul grid'5000, il est utilisé pour le stockage des données. Nous utiliserons cette même architecture pour toutes nos expériences.

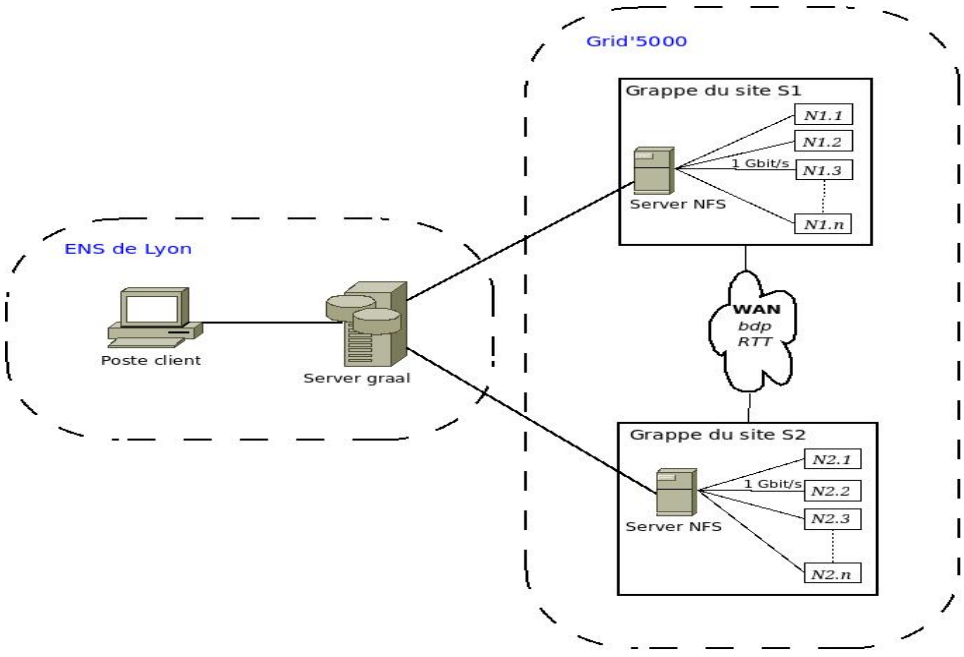

FIGURE 1.1 – Plate-forme expérimentale

À partir du poste client, nous allons transférer les fichiers sur le serveur graal. Une fois que les fichiers sont sur le serveur graal, nous allons les transférer par la suite au serveur NFS des différents sites avant de les migrer vers

1. `https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home`

les différents nœuds. Donc, nous allons faire des transferts de données point à point, c'est-à-dire à partir du serveur graal vers un serveur NFS, à partir du serveur graal vers un nœud, à partir du serveur NFS vers un nœud, etc. Ces transferts seront réalisés en utilisant les protocoles de transfert de données suivantes : Aspera [3], Bitspeed [6] et Expedat [7,8].

1.3 Problématiques et objectifs

La demande croissante pour l'échange rapide d'énormes quantités de données entre sites distants a conduit à l'émergence de nombreuses nouvelles solutions de transport de données qui promettent de transporter d'énormes quantités de données beaucoup plus rapide que les solutions FTP/TCP classiques. Actuellement, les solutions les plus courantes pour le transport de données fiable dans les réseaux IP sont basés sur le protocole TCP, qui a été développé dans les années 1970. Un certain nombre de documents décrivent comment TCP, avec quelques ajustements, peut fonctionner raisonnablement sur les réseaux locaux (LAN) avec une bande passante hautement disponible [20]. Toutefois, il est bien connu que TCP a un rendement très limité lorsqu'il est utilisé dans les réseaux longue distance avec une bande passante élevée - appelés "Long Fat Pipe Network (LFN)" [19]. Par exemple, un test avec iperf en utilisant l'architecture décrite dans la figure 1.1 sur une liaison de bout-en-bout de 1 Gbit/s avec un RTT de 50 ms (round-trip time) et en présence d'un taux de perte d'au moins 0,1% montre un débit de données d'environ 40 Mbit/s.

Il faut mentionner aussi que la plupart des WAN (Wide Area Network) actuels ne sont pas appropriés d'acheminer des téraoctets et pétaoctets de données vers la grille [19]. Tout d'abord, la grande latence du réseau longue distance implique des communications et des retransmissions de paquets perdus qui sont coûteuses. Ensuite, le débit disponible sur le lien d'accès à ce réseau est généralement inférieur à la somme des débits nécessaires si tous les processus communiquent en même temps sur ce lien. D'une manière générale les trois principaux challenges lorsque les Big Data [2] migrent vers la grille sont la localisation, la bande passante, et la qualité du réseau.

Plus la distance réseau entre le data center et le site de stockage d'origine est considérable, plus la latence est importante sur le WAN, et plus le transfert des données est long. Lorsque l'utilisation de la bande passante disponible n'est pas suffisante, les transferts de données prennent plus de temps à être effectuer. Compte tenu des contraintes inhérentes à TCP, l'augmentation de la bande passante à elle seule ne suffit pas à tenir l'objectif de transférer les big data avec le débit nécessaire. D'où l'émergence de nom-

2. Énormes volumes de données difficilement gérables avec des solutions classiques de stockage et de traitement.

breuses nouvelles solutions de transport de données qui peuvent transporter d'énormes quantités de données beaucoup plus rapide que les solutions FTP / TCP classiques.
Il est donc important de spécifier pour chaque modèle les caractéristiques du réseau modélisé. Le comportement des réseaux de grappe, par leur aspect dédié, bénéficie de modélisations [1,15,17,18] que nous détaillerons en chapitre 2.

L'objectif principal de ce stage est d'étudier en détail la gestion des grands volumes de données sur la grille de calcul et d'évaluer les capacités des solutions de transport dans les réseaux longue distance à haut débit.

1.4 Contribution

La contribution de ce stage s'articule en fonction des objectifs énoncés dans la section précédente.

☞ **Expériences et analyse des transferts de données**

Ce stage a permis de mener une étude sur les performances des protocoles de transfert de données. Cette étude propose de comparer 3 solutions : Aspera [3], Bitspeed [6] et Expedat [7][8] sur la grille. Ces expériences ont été menées sur les grappes du projet Grid'5000. Les travaux proposés établissent une relation entre le temps de transfert, la latence et le débit mesuré dû à l'utilisation optimale de la bande passante par chaque protocole. Pour atteindre cet objectif, nous avons effectué des transferts de données de taille variée afin de juger le comportement de ces protocoles dans le transfert de petit ou de gros fichier. Le chapitre 3 décrira le protocole expérimental mis en œuvre.

Chapitre 2

État de l'art

2.1 Les protocoles de transfert de données

L'objectif principal de notre travail est d'évaluer les capacités des solutions de transport dans un réseau WAN à grande vitesse. L'intérêt pour nous est le temps de transfert de données minimale possible de bout en bout sur de tels réseaux. Actuellement, il y a quelques différentes mesures de performance qui ont été utilisées pour évaluer ces déficiences en terme de temps de transfert dans les solutions open source et freeware. Par exemple, dans [22] Grossman et al. présentent l'évaluation de la performance de UDT [21] à travers un réseau de 10 Gbit/s. L'article montre comment en utilisant UDT et en présence de 116 ms de RTT, ce réseau a un débit maximal de 4,5 Gbit/s dans un seul flux de données. Deux flux parallèles réalise ainsi environ 5 Gbit/s et dans les 8 flux parallèles environ 6,6 Gbit/s sont atteints. En outre, un résultat de performance pour la transmission de données à l'aide de RBUDP a été présenté au 3ieme atelier international annuel de CineGrid [23]. Bien que la vitesse d'accès au disque limite la vitesse de transport de données à 3,5 Gbit/s, sur le lien entre Amsterdam et San Diego seulement 1,2 Gbit/s a été atteint. La distance de ce chemin est d'environ 10 000 km à travers la fibre optique, ce qui correspond à environ 100 ms de RTT.

Pour les solutions closed source commerciales, la situation diffère sensiblement. Il y a plusieurs publications des résultats de performance des solutions disponibles sur le marché, fournies par les fabricants eux-mêmes : Aspera [3], Expedat [7,8] et Bitspeed [6] - qui ont tous fait état des résultats parfaits. Toutefois, ces résultats fournissent principalement des informations commerciales pour attirer les clients potentiels et les conditions d'évaluation varient. Pour remédier à ce déficit, l'idée principale de notre travail est de placer toutes les solutions étudiées dans des conditions égales dans le même environnement.

2.1.1 Expedat

Expedat est une solution de transport de données basée sur UDP développée par Data Expedition Inc., USA. Le cœur de cette application comprend le Protocole Multi Transaction (MTP) [7,8], développé par le fondateur de Data Expedition, dans le but d'envoyer et de recevoir des fichiers plus rapidement et de manière fiable que les applications FTP traditionnelles. Expedat prend en charge beaucoup de plates-formes telle que Windows, Mac OSX, Linux/Solaris, NetBSD/FreeBSD, AIX et les plates-formes HP-UX. Selon le site Web de la société, Expedat permet la transmission de données avec 100% d'utilisation de la bande passante allouée et en présence de cryptage AES [8]. Il met en œuvre la logique de protocole de transport UDP sur un canal, et utilise un seul socket UDP sur chaque côté de la connexion pour la transmission des données et des informations de contrôle.

2.1.2 Bitspeed

Cette solution a été développée aux États-Unis. Il s'agit d'une application de transfert de fichiers basée sur le protocole TCP, et, selon le site Web du fournisseur [6], il permet d'utiliser pleinement la bande passante disponible. Bitspeed est également disponible avec un cryptage de données allant jusqu'à 24 Gbit/s et un cryptage AES allant jusqu'à 1600 Mbit/s. Les plates-formes supportées par Bitspeed sont Windows, Mac OSX, Linux et Solaris. Selon le mode d'emploi, cette solution adapte automatiquement ses paramètres avec les conditions du réseau et choisit les paramètres optimaux (débit, latence, etc.) pour la transmission de données.

2.1.3 Aspera

La technologie de transfert FASP de Aspera [3] est un logiciel innovant de la compagnie IBM qui élimine les goulots d'étranglement fondamentaux des technologies de transfert de fichiers classiques, tels que HTTP, FTP, et accélère les transferts sur des réseaux IP publics et privés. Cette approche permet d'améliorer le débit, indépendant de la latence du lien. En outre, les utilisateurs ont le contrôle sur les taux individuels de transfert et le partage de la bande passante, et une visibilité complète sur l'utilisation de la bande passante. Le temps de transfert de fichiers peut être garantie, indépendamment de la distance des points d'extrémités ou les conditions dynamiques du réseau, y compris les transferts sur les réseaux sans fil et les liaisons internationales fiables. Aspera intègre le cryptage des données, y compris l'authentification sécurisée du point d'extrémité, et la vérification de l'intégrité des données.

Comparaison et mesure :

Le tableau 2.1 compare les différents protocoles en terme des plates-formes supportées et le débit et la latence minimales et maximales mesurés lors de nos expériences.

	Plates-formes	**Débit Min**	**Débit Max**	**Latence Min**	**Latence Max**
Aspera	Windows, Linux, Mac OSX, Solaris, FreeBSD, Isilon OneFS	44.75 Mb/s	630 Mb/s	10 ms	23 ms
Expedat	Windows, Linux, Mac OSX, Solaris, NetBSD/FreeBSD, AIX, HP-UX	40 Mb/s	577 Mb/s	8 ms	19 ms
Bitspeed	Windows, Linux, Mac OSX, Solaris	19.4 Mb/s	304,5 Mb/s	5 ms	16 ms

TABLE 2.1 – Caractéristiques des protocoles

Pour évaluer la performance de ces protocoles, nous avons réalisé des transferts de données entre des paires de nœuds appartenant à deux grappes grid5000 différentes. Les expériences ont été réalisés sur les sites de Nancy et de Rennes.

Pour ce faire, on crée sur un nœud de Nancy 1440 fichiers de 6 Mo chacun rempli uniquement de zéros et un fichier vidéo d'une taille de 8 Go. Après on les transfert vers un nœud de Rennes.

Les expériences consistent à déterminer le temps mis pour transférer les 1440 fichiers et le fichier vidéo entre les deux nœuds. Les résultats sont présentés dans la figure 2.1.

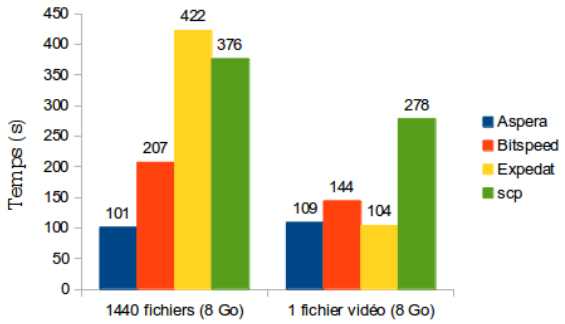

FIGURE 2.1 – Comparaison des protocoles de transfert

Il convient de relever trois points quant à l'analyse de cette figure :

- Certains protocoles se comportent mieux dans le transfert de grand fichier, mais très mauvais dans les petits fichiers. Comme c'est le cas pour le protocole Expedat.
- Certains protocoles se comportent très bien à la fois dans le transfert de petits fichiers et de grand fichier. Comme c'est le cas pour le protocole Aspera.
- On constate que certains protocoles comme celui d'Expedat offre un bon débit (efficace pour les gros fichiers), mais est plus faible côté latence (moins performant pour les petits fichiers).

2.1.4 Conclusion

Il existe bien des différences de performance entre les protocoles. De plus on constate une différence de performance en fonction de la façon d'utiliser le protocole. Il peut donc être intéressant de proposer des solutions dynamiques qui offrent des mécanismes permettant de choisir le bon protocole pour le bon usage.

2.2 Les modélisations de transfert de donneés

Il existe différents modèles et techniques qui ont été proposés pour mesurer et analyser la performance de transfert de données. En réalité, le transfert de données n'est pas seulement un problème des grilles de calcul, mais c'est aussi un problème qui concerne les réseaux de communication comme par exemple Internet. Dans cette partie nous allons présenter seulement les modèles que nous avons jugés proches de notre problématique.

2.2.1 Modèle de Hockney

Le modèle de Hockney [15] est historiquement un des premiers modèles de mesure de transfert de données, ce qui en fait l'un des modèles les plus utilisés. Pour calculer le temps d'un transfert de données, ce modèle introduit deux paramètres : la latence et la bande passante. La latence correspond au temps minimal de traversée du réseau, tandis que l'inverse de la bande passante correspond au taux de service maximum qu'offre le réseau. Ces deux paramètres ont été largement utilisés dans d'autres modèles. Le modèle de Hockney combine ces deux paramètres à travers une équation affine :

$$t(m) = L + m/B$$

Ainsi, le temps d'un transfert $t(m)$ qui envoie une quantité m de données est fonction de la latence L et de la bande passante B.

D'une manière plus formelle, le calcul de la latence L correspond à la durée d'envoi d'une quantité de données nulle. Elle se mesure en seconde. À l'inverse, la bande passante est le rapport entre une taille de données et sa durée de transfert. Elle correspond au débit et se mesure en octet par seconde. Pour obtenir les valeurs de ces deux paramètres il existe différents outils de mesure dont entre autre NetPIPE [16].

Un modèle similaire au modèle Hockney qui combine ces paramètres à travers une équation hyperbolique a été proposé. Ce modèle a ouvert la voie au concept de temps de latence proportionnelle à la taille de données. Ce concept sera repris dans les modèles de la famille LogP, en introduisant les concepts de surcoût logiciel (overhead).

2.2.2 Famille de modèle LogP

Le modèle de Hockney calcule les temps de transfert en fonction d'une simple équation affine et de deux paramètres. La famille des modèles LogP est apparue afin d'exprimer de manière détaillée les mécanismes intervenant dans un transfert. De ce fait, ces modèles sont plus complexes et augmentent le nombre de paramètres utilisés.

Dans le cas de communications point à point, la famille LogP comporte trois modèles : le modèle LogP [17], le modèle LogGP [1] et le modèle pLogP [18].

2.2.2.1 Modèle LogP

Le modèle LogP [17] est le modèle d'origine dont découlent les autres modèles de la famille LogP. Ce modèle définit quatre paramètres comme suit :

- L (Latency) : qui correspond à la latence réseau ;
- o (overhead) : le coût logiciel induit par le mécanisme de transfert ;
- g (gap) : le temps intrinsèque entre deux envois ou réceptions de paquets ;
- P (Processors) : le nombre de processeurs mis en jeu.

Trois de ces paramètres sont combinés par l'équation suivante :

$$2 \times o + L + \lceil \frac{k}{w} \rceil \times max(g, o)$$

Cette équation calcule le temps nécessaire à l'envoi de k octets divisés en w paquets. À la différence du modèle de Hockney, pour caractériser le taux fixe de transfert, le modèle LogP ajoute à la latence les coûts logiciels induits par l'émission et par la réception. Ces coûts correspondent à la création de paquets, l'encapsulation, etc. Un taux variable est associé aux différents paquets. Lorsque le modèle de Hockney précise une bande passante commune pour chaque taille de message, sans introduire la notion de paquets, le modèle LogP ajoute une contrainte entre paquets. Cette contrainte stipule que deux paquets consécutifs ne peuvent pas être transmis en moins de g unités de temps. Ce paramètre g représente la durée pendant laquelle le réseau transmet un paquet. Le modèle LogP donne des résultats efficaces pour des données de petite taille. Néanmoins, pour des données de grande taille les résultats sont moins précis. En effet, les valeurs de g et o sont identiques quel que soit la taille des données, et il paraît tout à fait logique que le surcoût logiciel est plus important pour de grandes données que pour de petites données. Pour remédier à cette perte de prédiction, le modèle LogGP a été introduit.

2.2.2.2 Modèle pLogP

Le modèle pLogP (parameterized LogP) [18] introduit une nouvelle façon de considérer les paramètres du modèle LogP. Ce modèle a pour paramètre

le surcoût logiciel et le gap en fonction de la taille des données. En outre, il distingue le surcoût logiciel en émission et en réception. Ainsi, le paramètre o devient $o_s(m)$ et $o_r(m)$, et le paramètre g s'écrit $g(m)$.

L'utilisation de communications bloquantes réduit l'impact et l'utilité des paramètres $o_s(m)$ et $o_r(m)$. Cette remarque, identifiée par les auteurs du modèle, a conduit à l'écriture de l'équation sous la forme :

$$L + g(m)$$

Cependant, pour des communications non-bloquantes il convient de réintroduire les paramètres $o_s(m)$ et $o_r(m)$.

Les auteurs vont plus loin que la simple expression du modèle. En effet, ils fournissent sur leur site Internet [1] un programme qui permet d'évaluer chaque paramètre en fonction de la taille des données. Ce programme propose plusieurs innovations. Par exemple, le gap pour de petites tailles de données est calculé en divisant le gap obtenu pour des données de grande taille par le temps d'aller-retour d'une donnée de taille nulle (RTT(0)). D'une manière similaire, les auteurs présentent une méthode détaillée pour la mesure des paramètres $o_s(m)$ et $o_r(m)$.

Ce modèle a deux principaux avantages : la flexibilité des paramètres o et g, en fonction de la taille des données, et l'existence d'un programme évaluant, pour un réseau donné, ces paramètres. Toutefois, il apparaît que ce modèle consiste principalement à obtenir des mesures depuis un programme qui teste le réseau. Il ne propose pas une équation pour déterminer $g(m)$. Pour répondre à ce problème, les auteurs de ce modèle proposent une correspondance entre les paramètres $o_s(m)$, $o_r(m)$ et $g(m)$ et les paramètres du modèle LogGP (o, g et G).

La famille de modèles LogP a été largement utilisée, modifiée et adaptée à différents problèmes. Certains auteurs ont par exemple introduit le nombre de processeurs comme paramètre dans des équations linéaires basées sur le modèle LogGP.

2.2.2.3 Modèle LogGP

Le modèle LogGP [1] est très similaire au modèle LogP. Ce modèle ajoute le paramètre G qui représente une valeur du gap en fonction de la taille des données. En d'autres termes, le gap G représente l'inverse de la bande passante. Avec ce nouveau paramètre l'équation du modèle précédent se réécrit par :

$$2 \times o + L + (k-1) \times G$$

1. `http://www.cs.vu.nl/albatross`

Nous noterons la grande similitude de cette équation avec celle du modèle de Hockney. Le modèle LogGP ne propose pas de grandes avancées à la fois dans la modélisation et dans la compréhension des mécanismes de transfert par rapport aux modèles précédents.

Il est aussi possible de représenter le modèle LogGP sous une forme parallèle, comme le montre le tableau 2.2.

LogP/LogGP	**pLogP**
L	$L + g(m) - o_s(m) - o_r(m)$
o	$(o_s(m) + o_r(m))/2$
g	$g(m)$
G	$g(m)/m$, pour une donnée m de grande taille
P	P

TABLE 2.2 – *Le modèle LogGP exprimé en fonction de pLogP*

2.2.3 Conclusion

En conclusion, le modèle de Hockney reste un modèle simple et pertinent, mais peu descriptif quant aux différents mécanismes utilisés lors d'une communication. Les modèles LogP et LogGP introduisent une plus grande description mais restent très similaires au modèle de Hockney au niveau des prédictions. Le modèle pLogP propose des paramètres dépendants de la taille des données. Cependant, le calcul du gap pour chaque taille de données est nettement pénalisant.

Deuxième partie

Chapitre 3

Expérimentations

3.1 Performances des protocoles de transfert

Dans le but de valider les protocoles de transfert, nous avons effectué des expériences sur la grille de calcul Française Grid'5000. Ces expériences ont été menées sur la plate-forme 1.1 décrit dans la section 1.2.

3.1.1 Transfert entre des nœuds de site différent

Dans cette expérience, nous avons réservé un nœud sur le site de Lyon (*taurus-3.lyon.grid5000.fr*) et un autre sur le site de Toulouse (*pastel-73.toulouse.grid5000.fr*). Pour effectuer les transferts, nous avons installé l'application serveur de chaque protocole sur le nœud de Lyon et l'application client de chaque protocole sur le nœud de Toulouse. La figure 3.1 présente les résultats obtenus. D'une manière générale, on constate que les protocoles commerciaux prennent moins de temps pour transférer les données par rapport au protocole scp.

3.1.2 Transfert entre des serveurs NFS de site différent

Dans cette expérience, nous avons accédé au serveur NFS sur le site de Lyon (*Lyon G5K*) et au serveur NFS sur le site de Toulouse (*toulouse G5K*). Pour transférer les données, nous avons installé l'application serveur et l'application client de chaque protocole sur les sites. La figure 3.2 présente les résultats obtenus. On peut constater que les protocoles commerciaux se comportent mieux par rapport au protocole scp.

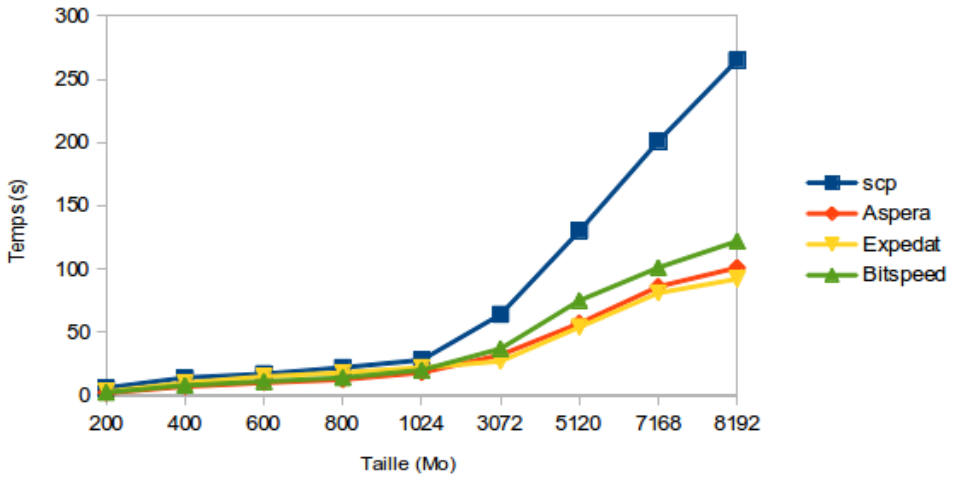

FIGURE 3.1 – Temps de transfert entre 2 nœuds de site différent

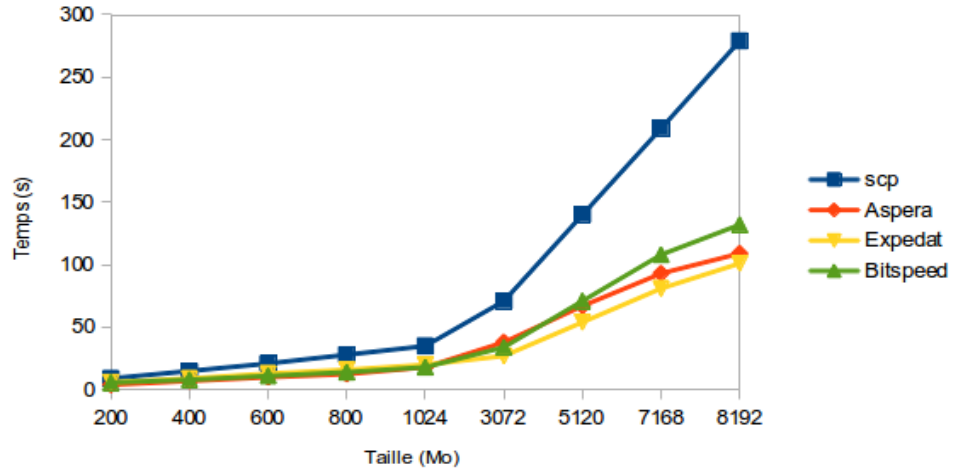

FIGURE 3.2 – Temps de transfert entre 2 serveurs NFS de site différent

3.1.3 Transfert entre un nœud et un serveur NFS de site différent

Dans cette expérience, nous avons réservé un nœud sur le site de Lyon (*taurus-3.lyon.grid5000.fr*) et accédé au serveur NFS du site de Toulouse (*toulouse G5K*). Pour effectuer les transferts, nous avons procédé de la même manière que dans l'expérience précédente, c'est-à-dire installer l'application serveur et client de chaque protocole sur chaque site. La figure 3.3 présente les résultats obtenus. On constate encore une fois que les protocoles commerciaux prennent moins de temps pour transférer les données par rapport

au protocole scp. Le protocole Aspera offre les meilleurs temps de transfert.

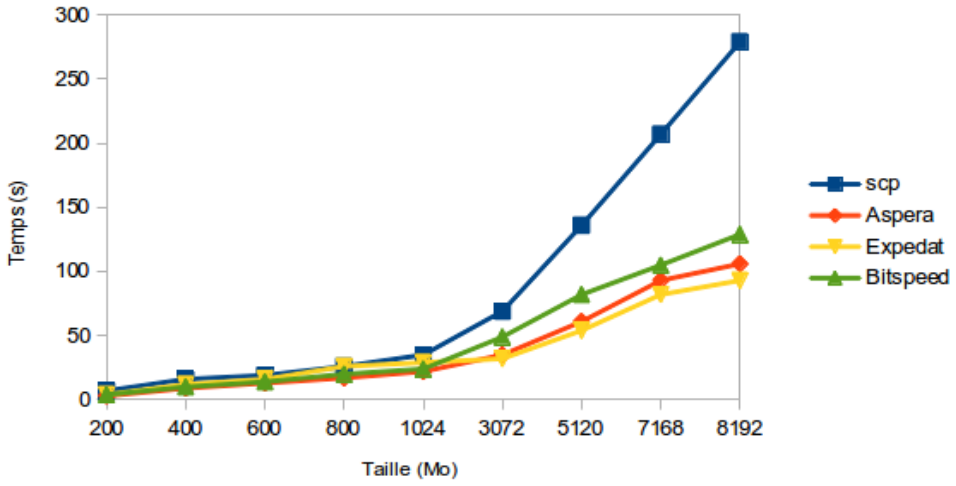

FIGURE 3.3 – Temps de transfert entre un nœud et un serveur NFS de site différent

3.1.4 Transfert entre le serveur graal et un serveur NFS

Dans cette expérience, nous avons accédé et installé sur le serveur graal (*graal.ens-lyon.fr*) l'application serveur et sur le serveur NFS du site de Lyon (*lyon G5K*) l'application cliente de chaque protocole. La figure 3.4 présente les résultats obtenus. On peut à nouveau remarquer que les protocoles Aspera, Expedat et Bitspeed offrent de meilleures performances par rapport au protocole scp.

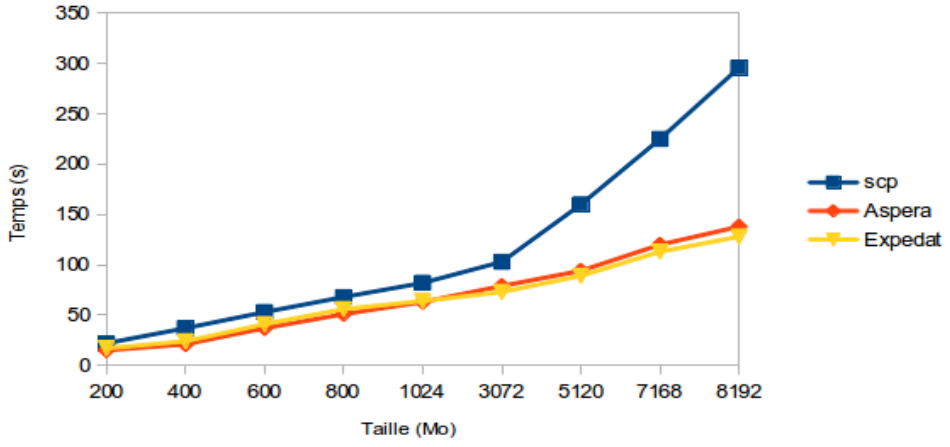

FIGURE 3.4 – Temps de transfert entre le serveur graal et un serveur NFS

3.2 Performances des modèles et techniques de mesure de transfert de donneés

Dans cette partie nous allons d'écrire les expériences réalisées pour mesurer les différents paramètres du modele LogGP, comme la mesure du *gap*, *l'overhead du temps d'envoi*, *l'overhead du temps de réception* et la *latence*. Ces expériences ont pour but de mettre en evidence le comportement réel du système et de présenter les premières analyses avant de construire le modèle mathématique qui sera présenté dans la partie suivante.

3.2.1 Description des expériences

Dans le chapitre 2, nous avons présenté les différents modèles mathématiques pour mesurer la performance en grille de calcul. À partir de l'étude théorique des modèles, nous avons décidé d'utiliser le modèle LogP [17] pour sa capacité à capturer des aspects qui permettent de décrire l'utilisation du réseau pendant le transfert de données.

Les expériences consistent en l'exécution d'un code appelé logp_multitest [1] dans l'environnement d'étude. Ce code fait parti d'un outil pour l'implémentation du modèle LogP par un système de passage de message qui s'appelle MPI.

Nous avons sélectionné deux sites Grid'5000 pour faire les mesures : Lyon et Toulouse. Sur chaque site nous avons fait des expériences sur l'environnement OAR pour l'utilisation des nœuds réservés. Nous avons fait des expériences pour le transfert de données de taille fixe.

Les tailles de données pour les expériences sont de 25 Mégaoctets, 50 Mo, 75 Mo, 100 Mo, 125 Mo, 150 Mo, 175 Mo, 200 Mo, 225 Mo et 250 Mo.

La figure 3.5 présente le modèle que nous avons utilisé. La latence L peut être observée comme le temps qui s'écoule entre l'envoi du premier bit d'une donnée de taille m depuis l'expéditeur P_0 jusqu'au récepteur P_1. $s(m)$ et $r(m)$ sont le temps d'envoi et de réception de la donnée quand les deux processeurs commencent leurs opérations simultanément. $s(m) = g(m)$ est le temps auquel l'expéditeur est prêt à envoyer la prochaine donnée. $r(m) = L + g(m)$ est le temps pendant lequel la donnée est reçue par le récepteur. $Os(m)$ et $Or(m)$ sont respectivement l'overhead d'envoi et de réception de la donnée de taille m. L'espace $g(m)$ est l'intervalle minimum de temps entre la transmission ou la réception de données consécutives.

1. `http://www.cs.vu.nl/albatross`

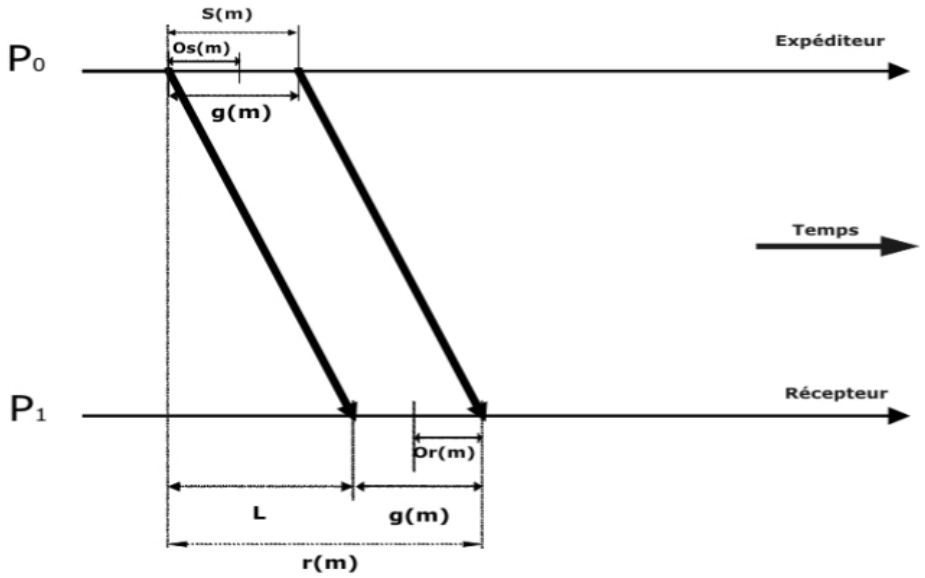

FIGURE 3.5 – Le modèle LogP

3.2.1.1 LogP/MPI 1.3

LogP/MPI [18] est un outil utilisé pour faire des mesures de benchmark en environnements distribués.

LogP/MPI évalue l'exécution des données envoyées et reçues pour des communications MPI. L'exécution est exprimée en terme de modèle paramétrisé de LogP, pour des données de diverses tailles. Les auteurs de LogP/MPI ont fournit une API pour rechercher des paramètres de LogP pour différentes tailles de données.

3.2.1.2 Logp_multitest

LogP_multitest est un programme utilisé pour faire des mesures, proposé par Luiz-Angelo Estefanel du Laboratoire ID-IMAG [2]. Le programme permet de connaître le comportement des paires de processeurs dans l'environnement pour envoyer et recevoir des données. Les options utilisées sont détaillées dans le fichier README du programme. Voici quelques une de ces options :

- Send : Indique qu'il utilisera un appel MPI Send.
- Recv : Indique l'utilisation d'un appel MPI Recv.
- -min-size : La plus petite taille de données à envoyer.

2. `http://www-id.imag.fr/Laboratoire/Membres/Estefanel_Luiz-Angelo`

- -max-size : La plus grand taille de données à envoyer.
- -o : Indique le fichier de sortie.

Il est important de noter que la taille de la mesure est exponentielle, par exemple, pour faire une mesure de 25 Mégaoctets comme taille minimale, il faut spécifier une taille de -min-size 26214400*26214400 .

3.2.1.3 Les commandes utilisées

Pour les expériences, la connexion aux nœuds Grid'5000 se fait par ssh, par exemple, pour la connexion sur taurus-11.lyon.grid5000.fr :

```
tpatrick@flyon: $ ssh root@taurus-11.lyon.grid5000.fr

Warning: Permanently added 'taurus-11.lyon.grid5000.fr,172.16.48.11'
(RSA) to the list of known hosts.
Linux taurus-11.lyon.grid5000.fr 2.6.32-5-amd64 1 SMP Mon Sep 23
22:14:43 UTC 2013 x86_64

   Squeeze-x64-base-1.8 (Image based on Debian Squeeze for
AMD64/EM64T)
Maintained by support-staff <support-staff@lists.grid5000.fr>

   Applications
* Text: Vim, nano
* Script: Perl, Python, Ruby
(Type "dpkg -l" to see complete installed package list)

   Misc
* SSH has X11 forwarding enabled
* Max open files: 65536

   More details: https://www.grid5000.fr/mediawiki/index.php/Squeeze-x64-base-1.8
root@taurus-11:
```

La commande principale pour exécuter le code est :

```
mpirun -np X logp_test -min-size T*T -max.size T*T -o resultat
```

X c'est la quantité de processeurs utilisés et T est la taille de la donnée.

Par exemple, pour une mesure avec 2 processeurs et une taille de donnée fixe de 25 Mégaoctets, on aura :

```
mpirun -np 2 logp_test -min-size 26214400*26214400 -max-size 26214400*26214400 -o
resultat
```

La sortie de cette commande, enregistrée dans le fichier résultat, contient les informations dans le tableau 3.6.

LogP network performance data: logp_test.Send.Recv								
Latency = 0.0000003								
Time	Bytes	Os	Os_min	Os_cnfint	Or	Or_min	Or_cnfint	g
1405257839	0	0.0000002	0.0000000	0.0000003	0.0000000	0.0000000	0.0000000	0.0000016
1405257839	26214400	0.0064491	0.0046711	0.0022057	0.0074044	0.0070278	0.0020937	0.0064540

FIGURE 3.6 – Performance réseau du modèle logP

La première ligne du tableau est la description de la commande, la deuxième la latence (en seconde) mesurée au moment de l'expérience. Les données sont organisées en colonnes : le temps (en microsecondes), la taille de données (en octets), l'overhead d'envoi (en seconde), l'overhead d'envoi minimal (en seconde), l'espace de temps minimal pour envoyer une donnée (en seconde), l'overhead de réception (en seconde), l'overhead de réception minimal (en seconde), l'espace de temps minimal pour recevoir une donnée (en seconde) et le gap (en seconde). Les données sont présentées en deux lignes, une avec une taille 0 et l'autre avec la taille demandée.

3.2.2 Les expériences sur taurus-11.lyon.grid5000.fr

Le tableau 3.1 présente les résultats obtenus lors de nos expériences pour la mesure des différents paramètres du modèle LogP. Les expériences ont été faites sur le site de Lyon dans les conditions décrites auparavant (section 1.2).

3.2.3 Les expériences sur pastel-73.toulouse.grid5000.fr

Le tableau 3.2 présente quant à lui les résultats obtenus lors de la mesure des différents paramètres du modèle LogP, pour les expériences sur le site de Toulouse. Ces expériences ont été faites aussi dans les conditions décrites auparavant (section 1.2).

Taille (Mo)	**Latence (s)**	**O_s (s)**	**O_r (s)**	**gap (s)**
25	0.0000003	0.0055190	0.0060305	0.0055236
50	0.0000003	0.0135005	0.0149702	0.0135062
75	0.0000003	0.0173047	0.0182720	0.0173094
100	0.0000003	0.0229525	0.0242307	0.0229562
125	0.0000003	0.0332497	0.0368666	0.0332550
150	0.0000003	0.0367356	0.0425598	0.0367407
175	0.0000003	0.0351907	0.0489063	0.0351950
200	0.0000003	0.0488382	0.0568850	0.0488432
225	0.0000003	0.0522441	0.0596211	0.0522480
250	0.0000003	0.0577335	0.0608290	0.0577387

TABLE 3.1 – Table des mesures sur le nœud de Lyon

Taille (Mo)	**Latence (s)**	**O_s (s)**	**O_r (s)**	**gap (s)**
25	0.0000012	0.0194218	0.0201255	0.0194273
50	0.0000012	0.0395902	0.0401559	0.0395960
75	0.0000012	0.0583565	0.0595662	0.0583614
100	0.0000012	0.0765779	0.0837329	0.0765828
125	0.0000012	0.0981575	0.1000983	0.0981627
150	0.0000012	0.1161653	0.1234202	0.1161697
175	0.0000012	0.1368746	0.1399666	0.1368794
200	0.0000012	0.1520239	0.1640336	0.1520291
225	0.0000012	0.1756973	0.1789241	0.1757024
250	0.0000012	0.1962725	0.2007752	0.1962779

TABLE 3.2 – Table des mesures sur le nœud de Toulouse

Chapitre 4

Modélisation

4.1 Modélisation du temps de transfert de données

L'objectif de cette partie est de formaliser un modèle mathématique du temps de transfert de données qui décrit le comportement observé, en accord avec le modèle paramétrisé LogGP et prendre les mesures.

4.1.1 Définition des variables

Les différentes variables utilisées dans le modèle sont définies dans le tableau 4.1 :

T_t	temps total de transfert des fichiers
T_{send}	temps total de transfert des fichiers vers le serveur NFS
T_{comput}	temps de génération de la vidéo au niveau du serveur NFS
T_{store}	temps de transfert de la vidéo vers le serveur de stockage
n	nombre de fichiers
$S(f_i)$	taille du fichier f_i
$BW_{link(host,storage)}$	bande passante disponible du lien entre le serveur NFS et le serveur de stockage
$Lat_{link(host,storage)}$	latence du lien entre le serveur NFS et le serveur de stockage
$O_s(f_i)$	surcoût en temps au processeur pour envoyer le fichier f_i
$O_r(f_i)$	surcoût en temps au processeur pour recevoir le fichier f_i
$O(f_i)$	surcoût en temps nécessaire au processeur pour recevoir et transmettre le fichier f_i
$g(f_i)$	intervalle de temps minimum entre deux transmissions consécutives sur un processeur
$L(f_i)$	latence du lien entre le nœud du fichier f_i et le serveur NFS
$R_{link(node,host)}$	débit du lien entre le nœud du fichier f_i et le serveur NFS
$G(f_i)$	Gap par octets pour les longs fichiers

TABLE 4.1 – Tableau des variables du modèle

4.1.2 Modèle

Considérons un ensemble de fichiers repartis sur plusieurs nœuds, notés f_i pour $i \in \{1, ..., n\}$. Considérons aussi un serveur NFS interconnecté avec les différents nœuds à l'aide d'un réseau Gb Ethernet. Dans un premier temps il faudra transférer tous les fichiers des nœuds vers le serveur NFS en prenant en compte toutes les caractéristiques du réseau comme le nombre de processeurs, la latence et le surcoût (overhead).

Selon le modèle LogGP, les caractéristiques d'un réseau N qui permet de transférer un fichier f_i d'un nœud vers le serveur NFS peut être formulé de la façon suivante :

$$\mathrm{N(f}_i)_p = (L(f_i), O(f_i), g(f_i), P) \tag{4.1}$$

À partir de la formulation ci-dessus on peut estimer le coût en temps nécessaire appelé T_{send} pour transférer les n fichiers des nœuds vers le serveur NFS avec cette formule :

$$\mathrm{T}_{send} = \sum_{i=1}^{n} L(f_i) + O_s(f_i) + O_r(f_i) + ((S(f_i) - 1) \times G(f_i)) \tag{4.2}$$

On considère que O est la moyenne d'overhead entre le temps d'envoi et de réception du fichier, donc :

$$\mathrm{O(f}_i) = \frac{O_s(f_i)+O_r(f_i)}{2} \tag{4.3}$$

D'où l'équation 4.2 devient :

$$\mathrm{T}_{send} = \sum_{i=1}^{n} L(f_i) + (2 \times O(f_i)) + ((S(f_i) - 1) \times G(f_i)) \tag{4.4}$$

En utilisant la relation qui existe entre la quantité des fichiers transférés notée $S(f_i)$ et la somme des temps d'attente entre paquets d'un même fichier notée $g(f_i)$, on peut calculer le débit de chaque lien par la formule suivante :

$$\mathrm{R}_{link(node,host)} = \frac{S(f_i)}{g(f_i)} \tag{4.5}$$

On peut estimer $G(f_i)$ en fonction du gap $g(f_i)$ et de la taille du fichier $S(f_i)$:

$$\mathrm{G(f_i)} = \frac{g(f_i)}{S(f_i)} \tag{4.6}$$

Après le transfert des fichiers, on génère la vidéo sur le serveur NFS. Le coût en temps nécessaire pour ce processus est appelé T_{comput}.

Une fois que la vidéo est générée, il faudra la transférer vers un serveur de stockage. Le coût en temps appelé T_{store} nécessaire pour le transfert est estimée en fonction de la taille de la vidéo S, de la bande passante et de la latence du lien notées respectivement $BW_{link(host,storage)}$ et $Lat_{link(host,storage)}$.

$$\mathrm{T}_{store} = \frac{S}{BW_{link(host,storage)}} + Lat_{link(host,storage)} \tag{4.7}$$

A partir des différents temps estimés ci-dessus, on peut formuler le temps total T_t de transfert des données des nœuds vers le serveur de stockage par la somme des temps de chaque étape de transfert.

$$\mathrm{T}_t = T_{send} + T_{comput} + T_{store} \tag{4.8}$$

D'où l'équation 4.8 devient :

$$\mathrm{T}_t = T_{comput} + \frac{S}{BW_{link(host,storage)}} + Lat_{link(host,storage)} + \sum_{i=1}^{n} L(f_i) + (2 \times O(f_i)) + ((S(f_i) - 1) \times G(f_i)) \tag{4.9}$$

4.1.3 Présentation des résultats

À partir des tables 4.2 , 4.3 et 4.4 il est possible de construire des graphiques pour analyser le temps de transfert des données calculé en fonction de la latence, de l'overhead, du gap par octet pour chaque taille de données et du débit. Les valeurs présentées dans les tableaux sont la moyenne des mesures prises pour chaque paramètre pendant les expériences.

Taille (Mo)	Latence (s)	O (s)	gap (s)	G (s)
25	0.00613907	0.00577475	0.0055236	2.11E-10
50	0.01233414	0.01423535	0.0135062	2.58E-10
75	0.01828667	0.01778835	0.0173094	2.20E-10
100	0.02443834	0.0235916	0.0229562	2.19E-10
125	0.03088884	0.03505815	0.033255	2.54E-10
150	0.0370115	0.0396477	0.0367407	2.34E-10
175	0.04271865	0.0420485	0.035195	1.92E-10
200	0.04893053	0.0528616	0.0488432	2.33E-10
225	0.05512317	0.0559326	0.052248	2.21E-10
250	0.05512317	0.05928125	0.0577387	2.20E-10

TABLE 4.2 – Table des paramètres pour taurus-11.lyon.grid5000.fr

Taille (Mo)	Latence (s)	O (s)	gap (s)	G (s)
25	0.02932845	0.01977432	0.0194323	7.41E-10
50	0.04920522	0.03987508	0.039599	7.55E-10
75	0.05946088	0.05896233	0.0683609	7.42E-10
100	0.07872334	0.0801265	0.0865879	7.30E-10
125	0.09645609	0.0991276	0.0981656	7.49E-10
150	0.12026263	0.11979268	0.1161697	7.38E-10
175	0.13675222	0.1384201	0.1368794	7.46E-10
200	0.1573376	0.15802945	0.1530456	7.25E-10
225	0.17700675	0.1773205	0.17670347	7.45E-10
250	0.19548234	0.19852476	0.1862743	7.49E-10

TABLE 4.3 – Table des paramètres pour pastel-73.toulouse.grid5000.fr

	Débit (Mb/s)	Latence (s)
Aspera	600	0.023
Expedat	500	0.019
Bitspeed	350	0.016

TABLE 4.4 – Table des paramètres pour les protocoles

4.1.3.1 Résultat obtenu pour le protocole Aspera

Les figures 4.1 et 4.2 présentent une comparaison entre le temps de transfert calculé à partir de notre modèle et le temps de transfert mesuré respectivement sur taurus-11.lyon.grid5000.fr et pastel-73.toulouse.grid5000.fr avec le protocole Aspera.

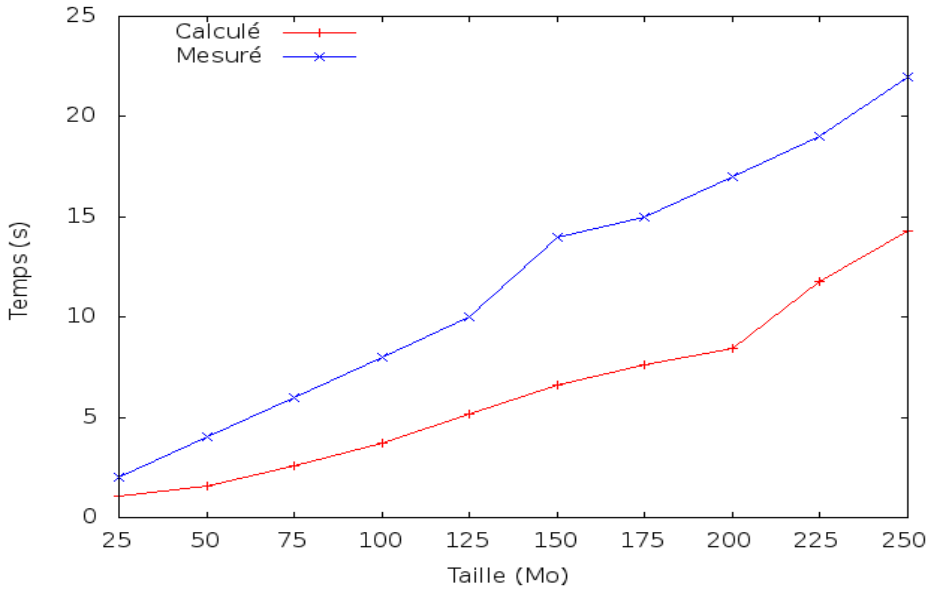

FIGURE 4.1 – Temps de transfert avec Aspera sur taurus-11.lyon.grid5000.fr

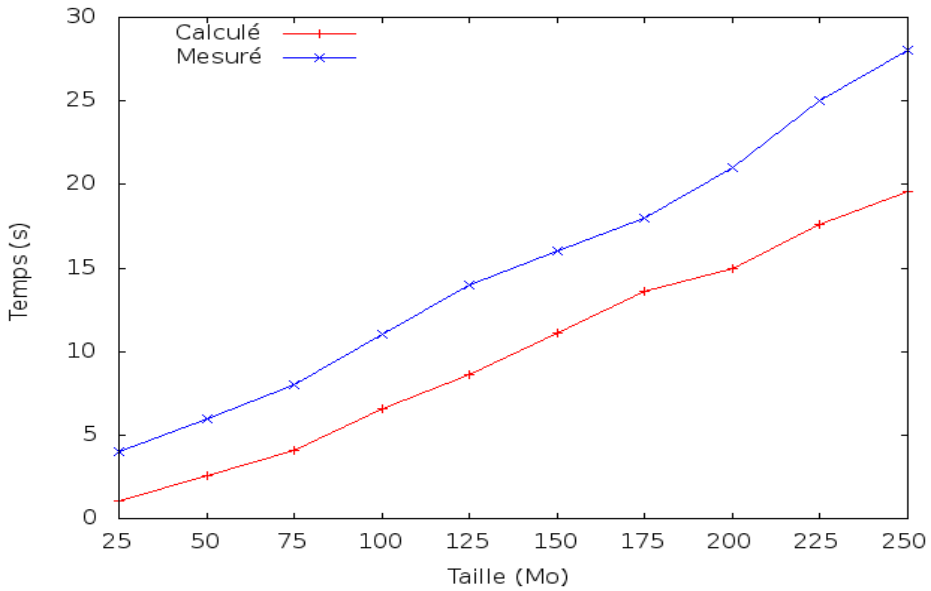

FIGURE 4.2 – Temps de transfert avec Aspera sur pastel-73.toulouse.grid5000.fr

4.1.3.2 Résultat obtenu pour le protocole Expedat

Les figures 4.3 et 4.4 présentent une comparaison entre le temps de transfert calculé à partir de notre modèle et le temps de transfert mesuré respectivement sur taurus-11.lyon.grid5000.fr et pastel-73.toulouse.grid5000.fr avec le protocole Expedat.

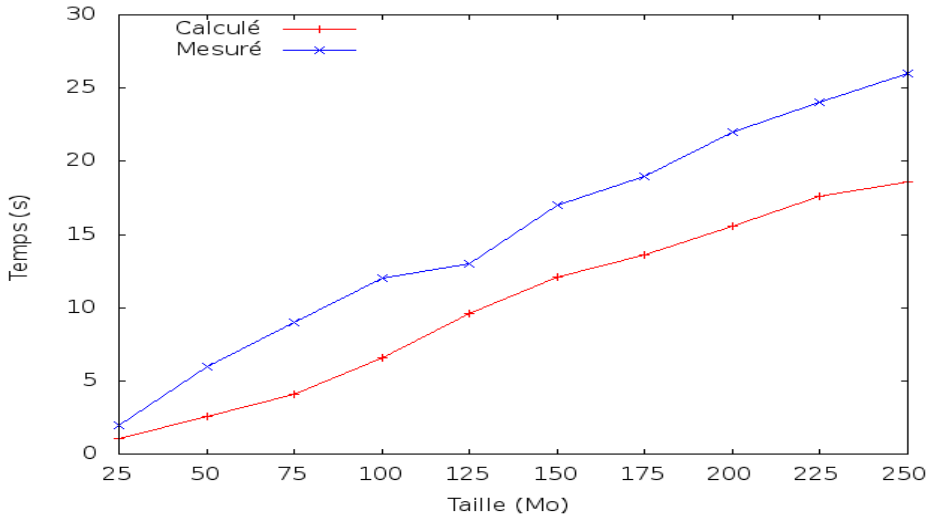

FIGURE 4.3 – Temps de transfert avec Expedat sur taurus-11.lyon.grid5000.fr

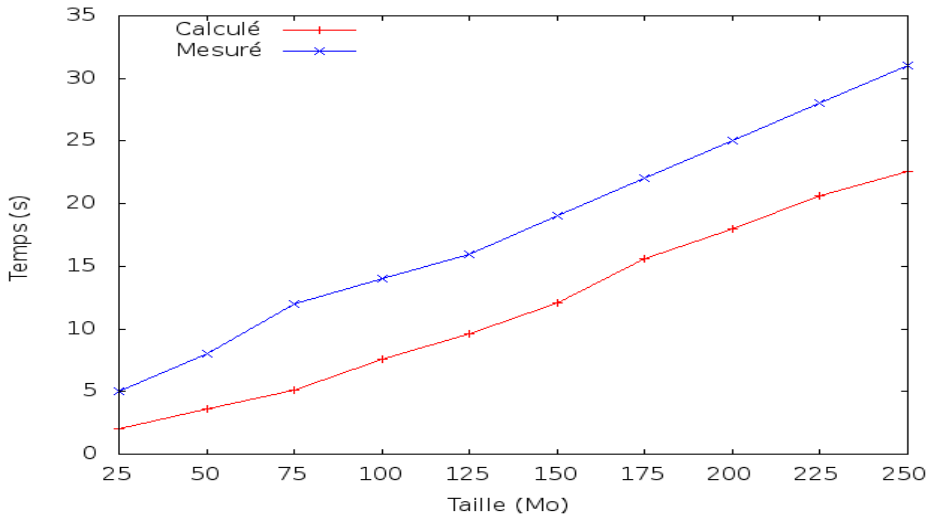

FIGURE 4.4 – Temps de transfert avec Expedat sur pastel-73.toulouse.grid5000.fr

4.1.3.3 Résultat obtenu pour le protocole Bitspeed

Les figures 4.5 et 4.6 présentent une comparaison entre le temps de transfert calculé à partir de notre modèle et le temps de transfert mesuré respectivement sur taurus-11.lyon.grid5000.fr et pastel-73.toulouse.grid5000.fr avec le protocole Bitspeed.

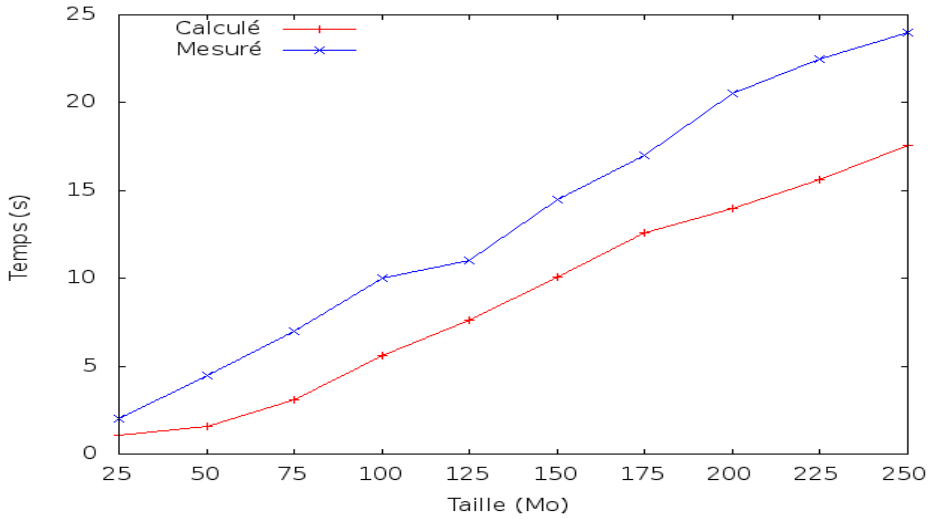

FIGURE 4.5 – Temps de transfert avec Bitspeed sur taurus-11.lyon.grid5000.fr

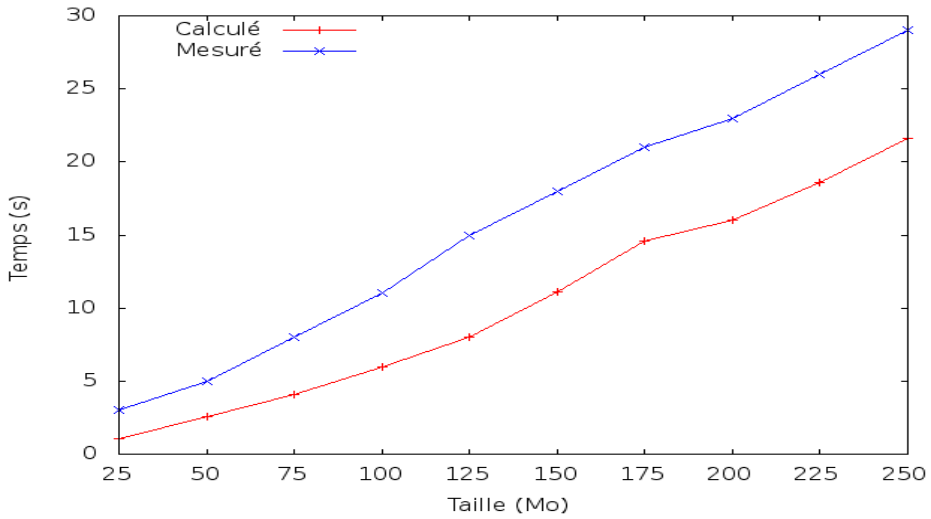

FIGURE 4.6 – Temps de transfert avec Bitspeed sur pastel-73.toulouse.grid5000.fr

Conclusion générale

Ce travail compare l'état de l'art des solutions commerciales pour le transport de données rapide et fiable par l'intermédiaire des réseaux longues distances (WAN) à haut débit. Le problème principal de ces recherches est que les sociétés vendeuses cachent souvent la technologie utilisée pour le transport de données accéléré. Le protocole utilisé dans la solution Expedat est couvert par des brevets américains. Toutefois, cela ne signifie pas que Expedat n'utilise pas n'importe quels algorithmes en plus de ceux décrits dans ces brevets. La seule méthode indépendante pour évaluer ces solutions commerciales est de les observer lors des évaluations dans des conditions bien définies.

Toutes les solutions étudiées se positionnent elles-mêmes comme des applications de transfert fiables à haute vitesse, conçus pour offrir des alternatives à FTP / TCP et surmonter les problèmes de performances de TCP sur des réseaux WAN à haut débit. Deux d'entre eux, Expedat et Aspera utilisent les sockets UDP et mettre en œuvre les logiques de protocole de niveau utilisateur, et Bitspeed exploite la pile TCP du système d'exploitation Linux.

Les résultats obtenus montrent que les solutions basées sur le protocole TCP héritent ses problèmes sur les liens à haut débit, nous avons constaté une diminution significative des taux de données jusqu'à 27% de la capacité du lien. Cependant, les solutions basées sur le protocole UDP montrent une bonne utilisation des liens à haut débit, nous avons donc constaté une utilisation des taux de données jusqu'à 60% de la capacité du lien, même en présence de RTT jusqu'à 100 ms. La comparaison a montré que la durée la plus faible de transfert de chaque solution est assez proche de l'idéal, et que la différence des valeurs de sortie obtenues sont proches de la réalité pour toutes les solutions.

Bibliographie

[1] Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman. *LogGP : Incorporating Long Messages into the LogP Model*. Proceedings in 7th Annual ACM Syposium on Parallel Algorithms and Architectures, Santa Barbara, CA, E.U.A. 1995.

[2] Y. Wu, S. Kumar, and S.-J. Park. *Measurement and performance issues of transport protocols over 10 Gbps high-speed optical networks*. Computer Networks, vol 54. 2010, pp. 475-488. doi :10.1016/j.comnet.2009.09.017.

[3] Aspera. *Custumer Deluxe Digital Studios*. [retrieved : 11, 2012]. http ://asperasoft.com/customers/customer/view/Customer/show/deluxe-digital-studios/.

[4] Y. Gu and R. L. Grossman. *UDP-based datatransfer for high-speed wide area networks*. Computer Networks. Austin, Texas, USA. May 2007, Vol. 51, issue 7, pp. 1465-1480.

[5] E. He, J. Leigh, O. Yu, and T. A. DeFanti. *Reliable Blast UDP : Predictable High Performance Bulk Data Transfer*. Proc. of IEEE Cluster Computing. Chicago, USA. Sept. 2002, pp. 317-324.

[6] Bitspeed LLC. *From Here to There - Much Faster*. White-paper. [retrieved : 10, 2012.] http ://www.bitspeed.com/wp-content/uploads/2011/10/BitSpeed-White-Paper-From-Here-to-There-Much-Faster.pdf.

[7] Data Expedition, Inc. Data Expedition. *Difference*. [retrieved : 10, 2012] http ://www.dataexpedition.com/downloads/DEI-WP.pdf.

[8] Data Expedition, Inc. Overview. Data Expedition, Inc. [retrieved : 10, 2012.] http ://www.dataexpedition.com/expedat/.

[9] B. W. Settlemyer, N. S. V. Rao, S. W. Poole, S. W. Hodson, S. E. Hick, and P. M. Newman. *Experimental analysis of 10Gbps transfers over physical and emulated dedicated connections*. Proc. of Computing, Networking and Communications (ICNC). Maui, Hawaii, USA. 2012, pp. 845-850.

[10] X. Wu and A. A. Chien. *Evaluation of rate-based transport protocols for lambda-grids*. High performance Distributed Computing, 2004. Proc. of 13th IEEE International Symposium on High Performance Distributed Computing.Honolulu, Hawaii, USA. 2004, pp. 87-96.

[11] S. Höhlig. *Optimierter Dateitranfer über 100 Gigabit/s*. 100 Gigabit/s Workshop of the DFN, Mannheim. Sept. 2011.

[12] Matti Vanninen, James Z. Wang. *On Benchmarking Popular File Systems*. Clemson University Study (2009).

[13] Hongzhang Shan, Katie Antypas, John Shalf. *Characterizing and Predicting the I/O Performance of HPC Applications Using a Parameterized Synthetic Benchmark*. CRD/NERSC, Lawrence Berkeley National Laboratory Berkeley, CA 94720.

[14] Ningning Zhu, Jiawu Chen, Tzi-cker Chiueh, Daniel Ellard. *An NFS Trace Player for File System Evaluation*. Technical Report TR-14-03, Harvard University, December 2003.

[15] Hockney, R. *Performance Parameters and Benchmarking of Supercomputers*. Parallel Computing, Volume 17, 1991, pages 1111-1130.

[16] Quinn O. Snell, Armin R. Mikler, John L. Gustafson. *NetPIPE : A Network Protocol Independent Performance Evaluator*. In : IASTED Internation Conference on Intelligent Information Management and Systems, (1996).

[17] Culler, D., Karp, Patterson, D., Sahay, A., Schauser, K., Santos, E., Subramonian, R., Von Eicken, T. *LogP : Towards a Realistic Model of Parallel Computation.* Proceedings in Four ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, San Diego, C.A. E.U.A. 1993.

[18] Kielmann, T., Bal, H., Verstoep, K. *Fast measurement of LogP parameters for message passing platforms.* In : Rolim, J.D.P. (ed.) IPDPS Workshops, Cancun, Mexico. Lecture Notes in Computer Science, vol. 1800, pp. 1176–1183. Springer-Verlag, London (2000).

[19] H. Kamezawa, M. Nakamura and M. Nakamura. *Inter-Layer Coordination for Parallel TCP Streams on Long Fat Pipe Networks.* Proc. of the 2004 ACM/IEEE conference on Supercomputing. Pittsburg, PA, USA. 2004, pp. 24 -34.

[20] Y. Wu, S. Kumar, and S.-J. Park. *Measurement and performance issues of transport protocols over 10 Gbps high-speed optical networks.* Computer Networks, vol 54. 2010, pp. 475-488. doi :10.1016/j.comnet.2009.09.017

[21] Y. Gu and R. L. Grossman. *UDP-based datatransfer for high-speed wide area networks.* Computer Networks. Austin, Texas, USA. May 2007, Vol. 51, issue 7, pp. 1465-1480. doi :10.1016/j.comnet.2006.11.009

[22] R. L. Grossman, Y. Gu, X. Hong, A. Antony, J. Blom, F. Dijkstra, and C. de Laat. *Teraflows over Gigabit WANs with UDT.* Journal of Future Computer Systems. Volume 21, 2005, pp. 501-513. doi :10.1016/j.future.2004.10.007

[23] L. Herr and M. Kresek. *Building a New User Community for Very High Quality Media Applications On Very High Speed Networks.* CineGrid. [retrieved : 02, 2013] http ://czechlight.cesnet.cz/documents/publications/network-architecture/2008/krsek-cinegrid.pdf.